Napredni razvoj programske potpore za web

- predavanja -

2022./2023.

---

5. HTTP/2

Creative Commons

- slobodno smijete:
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **prerađivati** djelo
- pod sljedećim uvjetima:
  - **imenovanje:** morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno:** ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima:** ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela.
Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.
Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava. *Tekst licence preuzet je s http://creativecommons.org/*

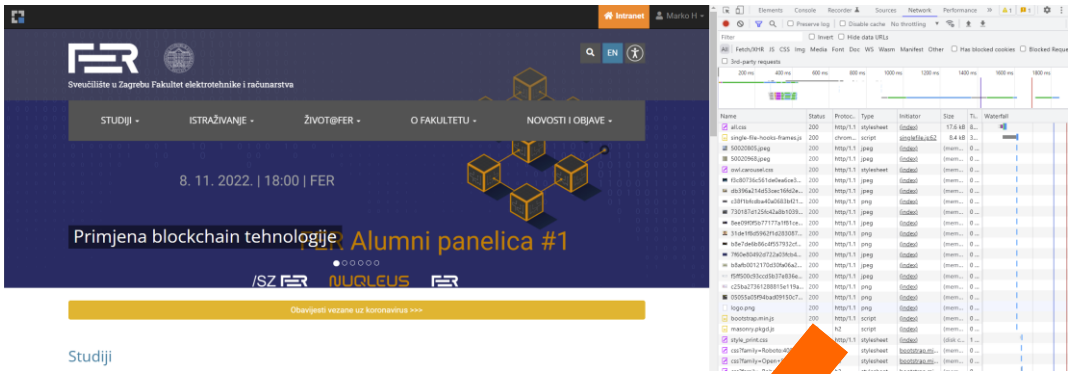
Uvod

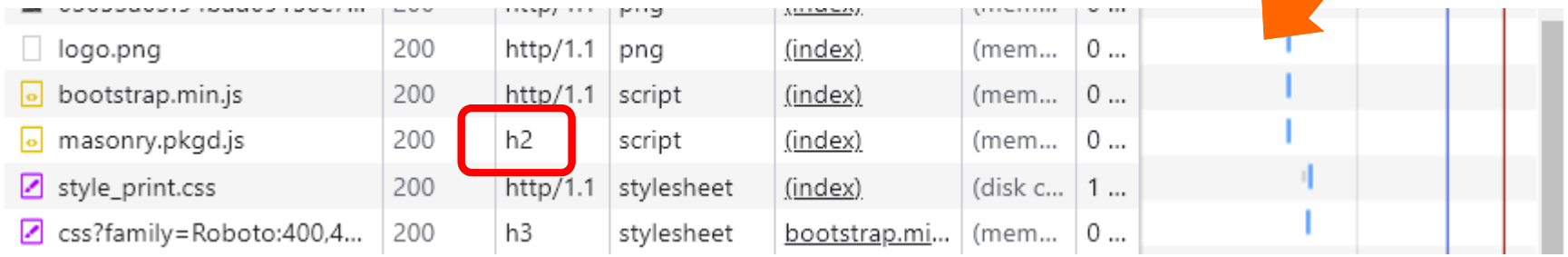
- Objavljen kao IETF specifikacija RFC 7540 (https://www.rfc-editor.org/rfc/rfc7540, 2015.)
  - Cilj pri izradi HTTP/2 protokola bio je otkloniti mnoge nedostatke HTTP/1.1 te povećati učinkovitost i sigurnost.
  - Glavna razlika između HTTP/2 i HTTP/1.1 verzija HTTP protokola je u načinu na koji se podaci grupiraju u okvire i prenose između klijenta i poslužitelja.

- Značajne razlike HTTP/2 u odnosu na HTTP/1.1:
  - **HTTP/2 je binarni protokol**, dok HTTP/1.1 podatke prenosi kao niz znakova
  - omogućeno je **multipleksiranje više tokova podataka** (*data streams*) preko jedne konekcije: klijent može poslati više zahtjeva, a poslužitelj može odgovarati u bilo kojem redoslijedu, odnosno kad odgovori postanu spremni
  - Uvedena je **kompresija zaglavlja poruka** koja uvelike sprječava redundantnost zaglavlja koja je prisutna kod ranijih verzija HTTP protokola
  - ***server push***
    - Poslužitelj može unaprijed poslati resurse iako ih klijent nije eksplicitno zatraži:. poslužitelj anticipira resurse koje će klijent zatražiti te tako štedi *Round Trip Time* (RTT)
  - **prioritizacija zahtjeva**
  - **kontrola toka** na razini veze i na razini *streama*
  - HTTP/2 **definira nekoliko vrsti podatkovnog okvira** (*data frame*):
    - DATA, HEADERS, PRIORITY, RST_STREAM, SETTINGS, PUSH_PROMISE, PING, GOAWAY, WINDOW_UPDATE, CONTINUATION

Provjera kompatibilnosti preglednika

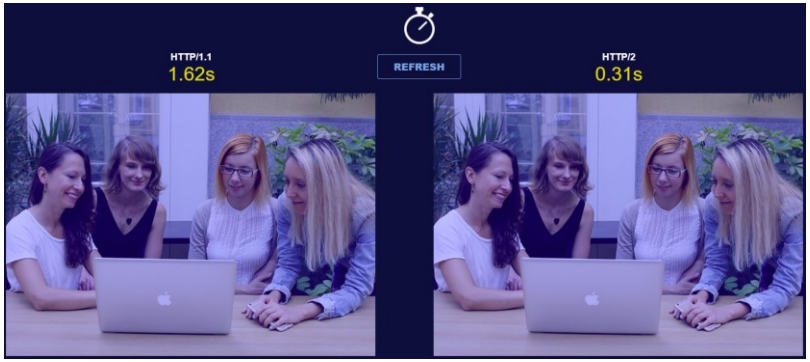
- Kako provjeriti da web poslužitelj podržava HTTP/2?
  - Otvoriti konzolu pretraživača (Chrome: CTRL+SHIFT+I)
  - Učitati HTTPS adresu sjedišta
  - U kartici „Network” potražiti redak s oznakom protokola „h2”
  - Npr. za https://www.fer.unizg.hr/
  - Prethodno uključiti logiranje (Chrome: Record Log, CTRL+E)
  - Snimiti log (*.har)

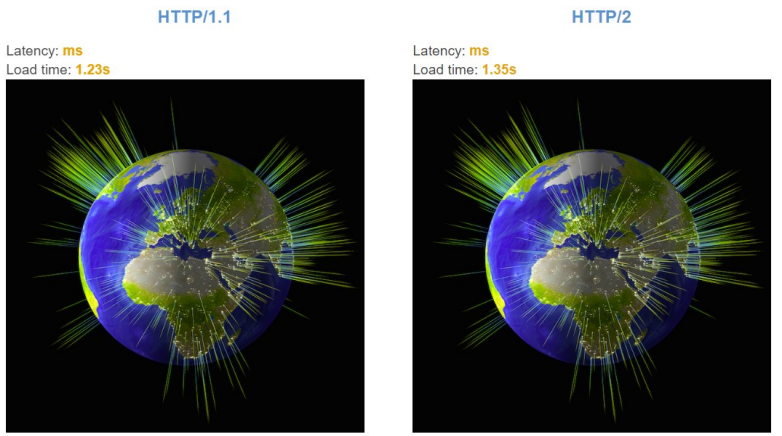




Usporedba perfomansi

- HTTP/2 usporedno s HTTP/1.1 najbolje pokazuje performanse u slučajevima:
  - Prijenosa velikog broja datoteka
  - Ponovnog prijenosa istih datoteka
    - Prema istom klijentu
    - Prema različitim klijentima
  - Prijenosa datoteka koje se mogu uspješno binarno komprimirati (npr. tekstne datoteke)





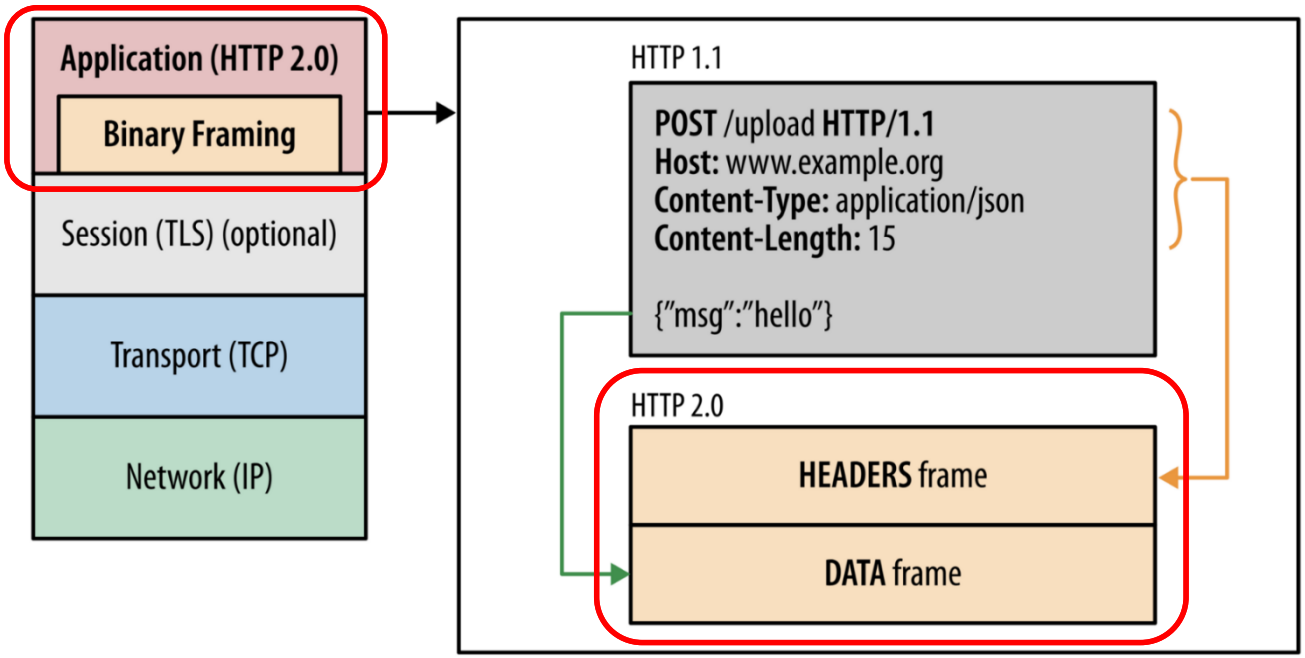
- HTTP/2, kao i svi HTTP protokoli, ne pamti stanje (*stateless*)

Jednostavna usporedba performansi HTTP/1.x i HTTP/2

http://www.http2demo.io/
https://http2.akamai.com/demo

Binarni prijenos podataka

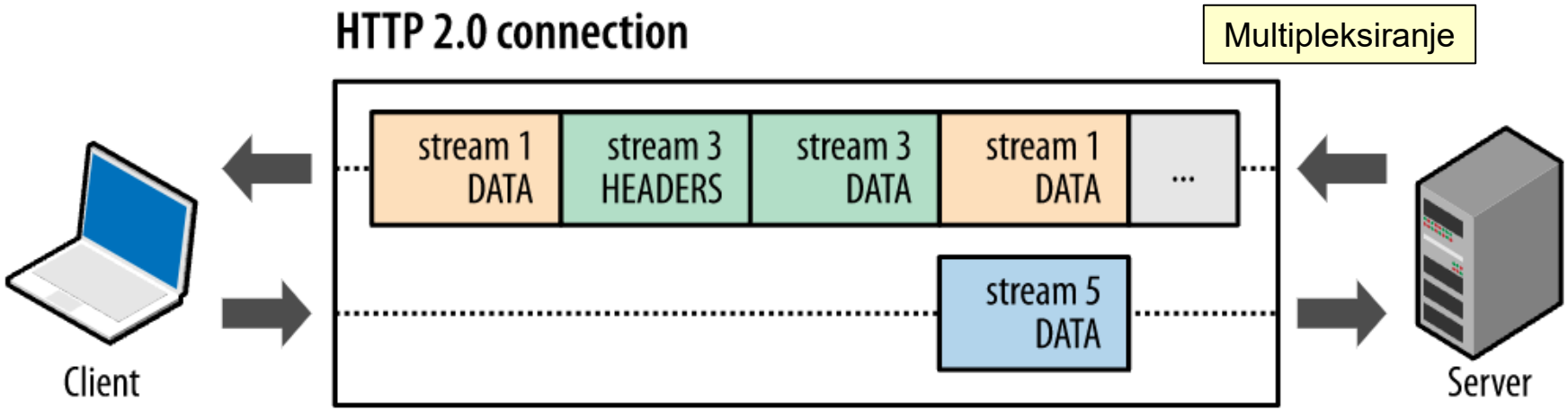
- U središtu svih poboljšanja performansi HTTP/2 je uvođenje novog **binarnog sloja uokvirivanja** (*binary framing layer*) iznad TCP/IP (obaveznih) i TLS (opcionalnog) protokola, koji određuje kako se HTTP poruke formatiraju i prenose između klijenta i poslužitelja.



https://web.dev/performance-http2/

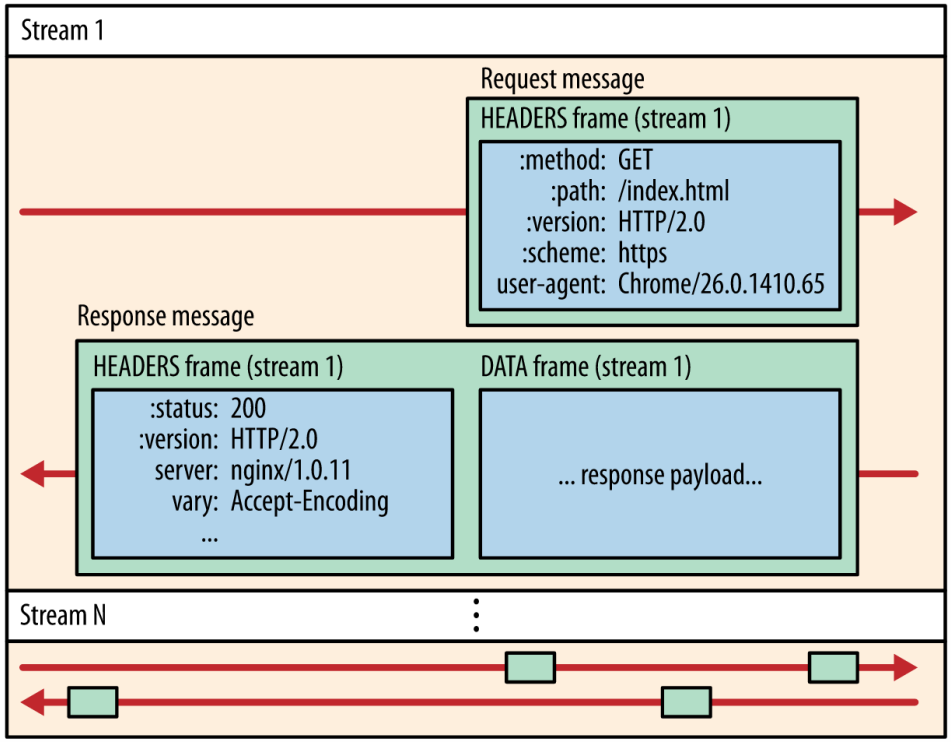
HTTP/2 konekcija

- Uvođenjem **binarnog uokvirivanja** (*binary framing*) mijenja se način razmjene podataka između klijenta i poslužitelja
- **Osnovni elementi strukture HTTP/2 konekcije**:
  - **Tok** (*stream*): Dvosmjerni (bidirekcionalni) tok podataka kroz uspostavljenu vezu koji sadržava barem jednu poruku.
  - **Poruka** (*message*): Cjeloviti niz okvira koji mapiraju logički zahtjev i poruku odgovor.
  - **Okvir** (*frame*): Najmanji element komunikacije HTTP/2 protokolom. Sadržava zaglavlje okvira (*frame header*) i identifikator toka (*stream ID*) kojemu okvir pripada. Sadržava binarne podatke.



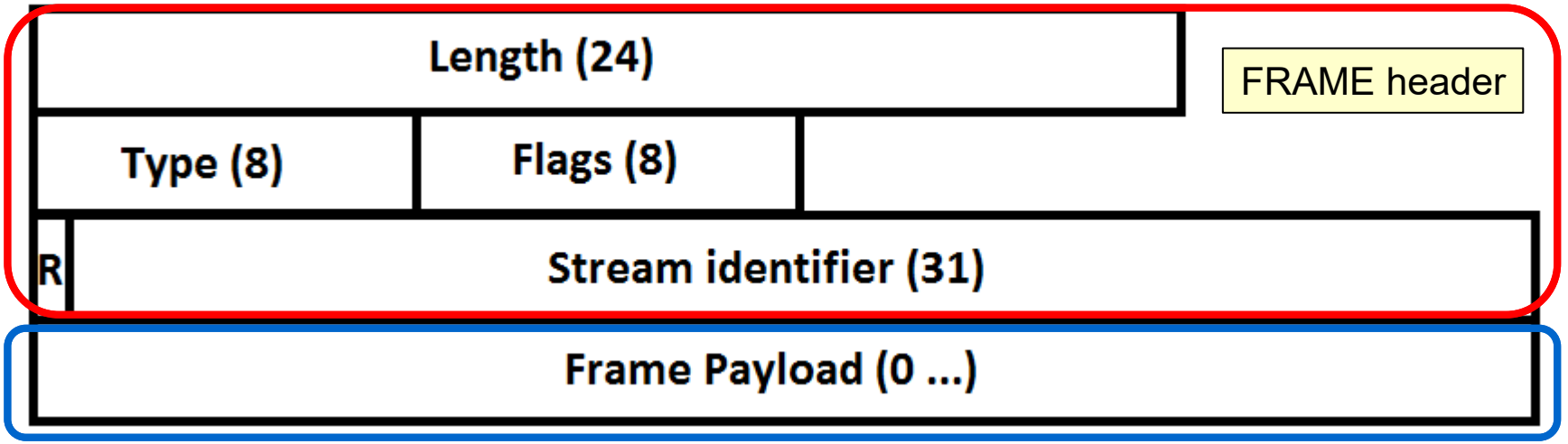
Tokovi, poruke i okviri

- Svaki HTTP/2 zahtjev i odgovor dobivaju zajednički **jedinstveni ID** koji se naziva **ID toka** (*stream ID*), a HTTP/2 zahtjev i odgovor podijeljeni su u okvire. Okviri predstavljaju binarne podatke (npr. HTTP zaglavlja, tekst, poruke).
  - *Stream* ID koristi se za identifikaciju kojem zahtjevu ili odgovoru okvir pripada.
  - **Tok podataka** (*data stream*) **je zbirka okvira s istim ID-om *streama***.
- Za postavljanje HTTP/2 zahtjeva, klijent prvo dijeli zahtjev na binarne okvire i okvirima dodjeljuje ID streama zahtjeva. Zatim započinje TCP/IP vezu s poslužiteljem. Nakon toga klijent šalje okvire poslužitelju. Nakon što poslužitelj ima spreman odgovor, on dijeli odgovor na okvire i daje okvirima odgovora **isti ID *streama***. Potom poslužitelj nastavlja slati odgovor u okvirima.
  - ID *streama* je neophodan jer se više zahtjeva izvoru šalje putem jedne TCP veze pa ID omogućuje identifikaciju kojem zahtjevu ili odgovoru okvir pripada.



Binarni okviri

- Nakon uspostave konekcije, klijent i poslužitelj komuniciraju razmjenom binarnih okvira
- Svi okviri imaju zaglavlje (*FRAME header*) dužine 9 bajta:
  - **Length**: dužina okvira = 24 bit
    - Jedan okvir može prenijeti do 2^{24} bitova (~16MB) podataka, ipak zbog optimizacije veličina je do 2^{14} , a veći okvir klijent i poslužitelj moraju dogovoriti (SETTINGS_MAX_FRAME_SIZE)
  - **Type**: tip i semantika okvira, nepoznati će biti odbačeni = 8 bit
  - **Flags**: boolean zastavice = 8 bit
  - **R**: Rezervirano = 1 bit (uvijek vrijednost 0)
  - **Stream identifer**: jedinstveni identifikator HTTP/2 toka = 31 bit



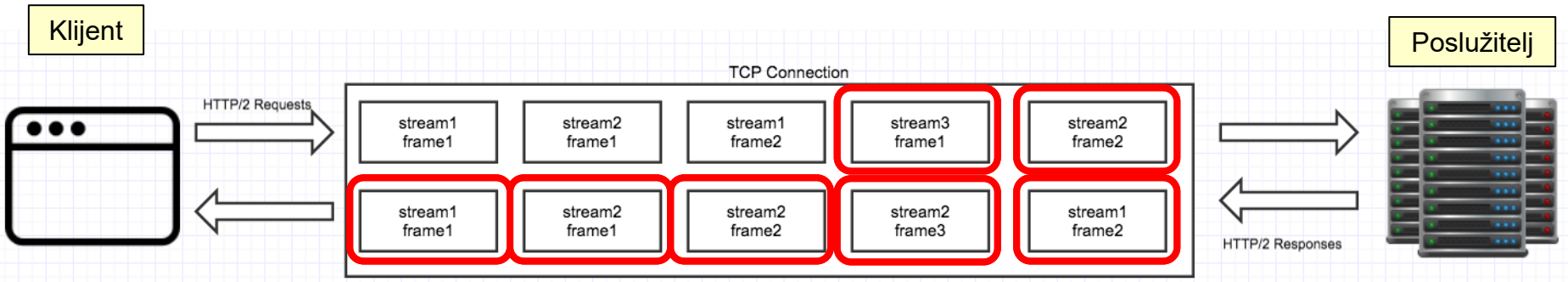
HTTP/2 multipleksiranje (1)

- Ako klijent koji podržava samo HTTP/1.x želi ostvariti višestruke paralelne zahtjeve kako bi povećao performanse, onda mora otvoriti višestruke TCP konekcije
  - Ova neučinkovitost je posljedica HTTP/1.x ***response queuing*** svojstva gdje se u jednoj konekciji odgovori serijaliziraju (FIFO) samo jedan odgovor može biti isporučen odjednom, unutar jedne konekcije.
  - ***Head-of-line blocking*** zbog prekoračenja maksimalnog broja konekcija jednog klijenta i poslužitelja.

- Klijent koji podržava HTTP/2 ne podliježe ovim ograničenjima
  - *Binary framing layer*
  - Uvodi se multipleksiranje poruka zahtjeva i odgovora. Jedna HTTP/2 poruka podijeljena je na međusobno neovisne okvire koji se mogu asinkrono slati i primati. Demultipleksiranje kod klijenta.

HTTP/2 multipleksiranje (2)

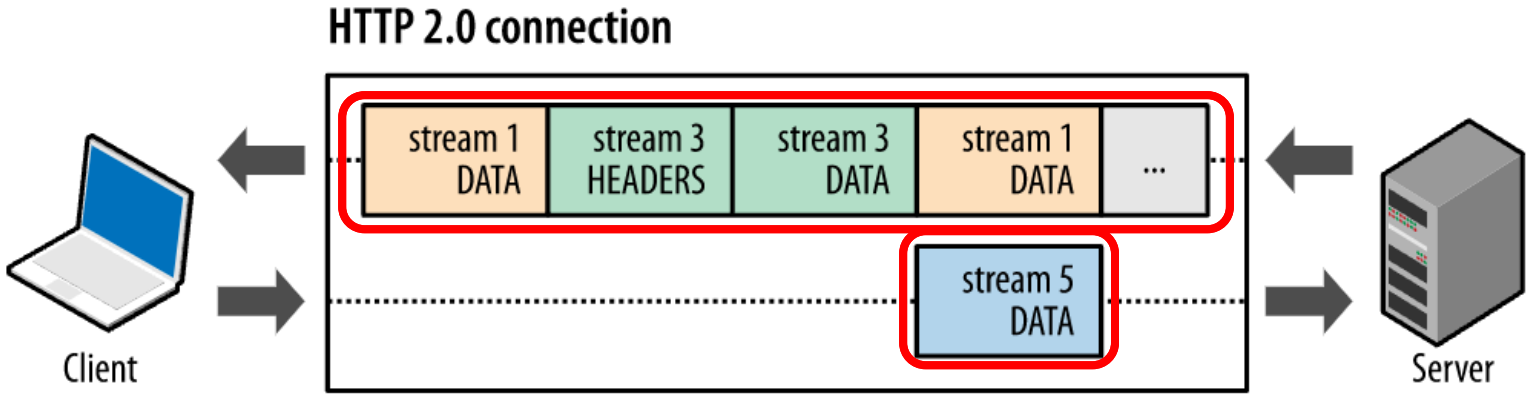
- Jedna TCP veza može se koristiti za slanje HTTP zahtjeva samo jednom poslužitelju
  - Za povezivanje s više poslužitelja potrebno je više TCP veza
- Svi HTTP/2 zahtjevi obavljaju se putem uspostavljene TCP veze
  - Više HTTP/2 zahtjeva podijeljeno je u okvire i dodijeljeni su im odgovarajući ID-ovi toka.
  - **Svi okviri iz više tokova šalju se asinkrono. Poslužitelj također šalje odgovore asinkrono.** Ako jedan odgovor predugo traje, drugi ne moraju čekati da završi. Zahtjev i odgovor događaju se paralelno, dok klijent šalje okvire poslužitelj također šalje okvire natrag klijentu.
  - Klijent prima okvire sa poslužitelja i raspoređuje ih prema ID *streama*.



Veza prema određenoj domeni ostane otvorena još neko vrijeme nakon prestanka zahtjeva

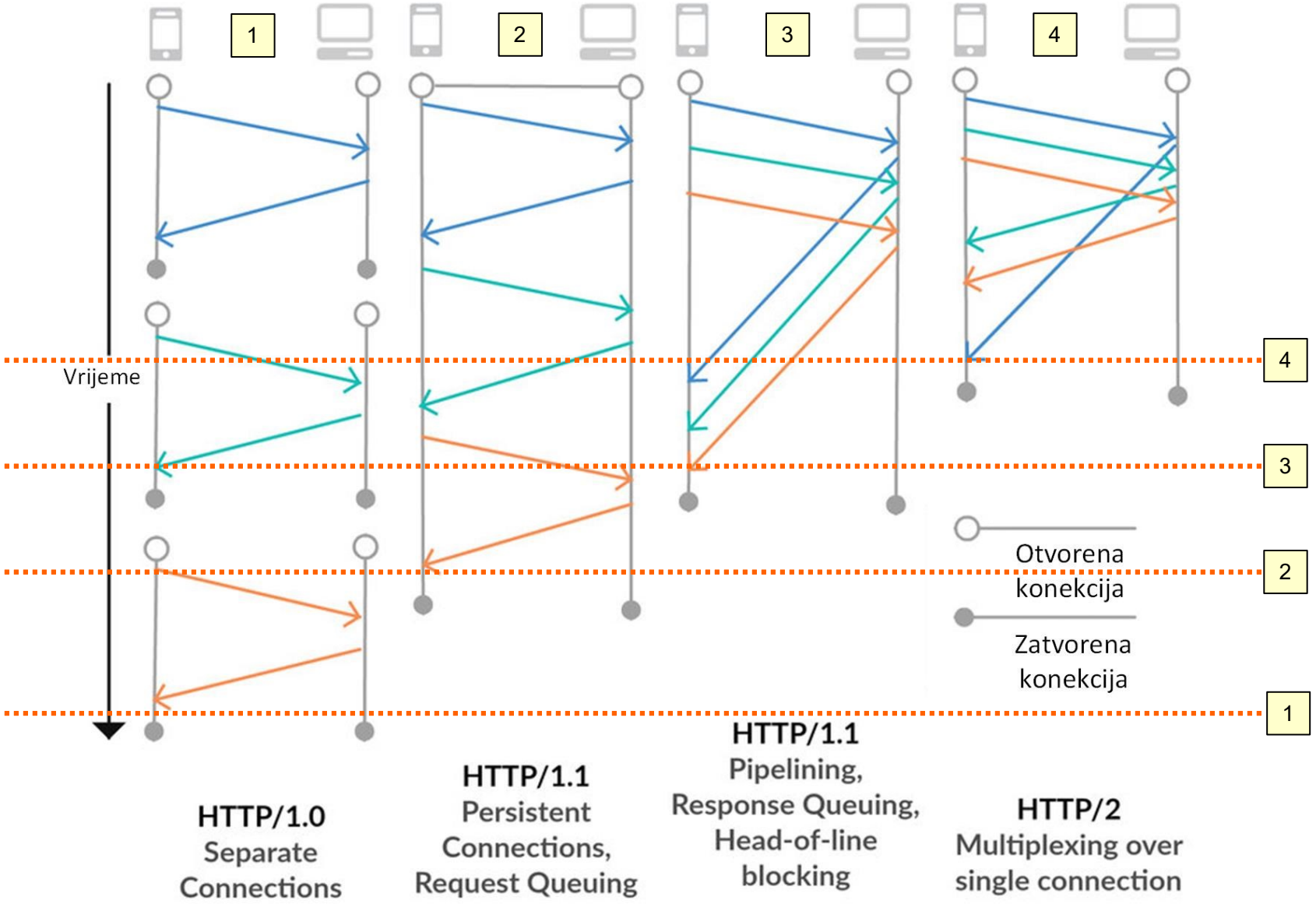
HTTP/2 multipleksiranje (3)

- U ovom primjeru klijent šalje DATA okvir (*stream* 5) poslužitelju
- Istodobno poslužitelj odgovara sa multipleksiranim nizom okvira za *streamove* 1, 2 i 3
- Nema čekanja na zahtjev, čekanja poruke, blokiranja i obaveznog FIFO redoslijeda poruka



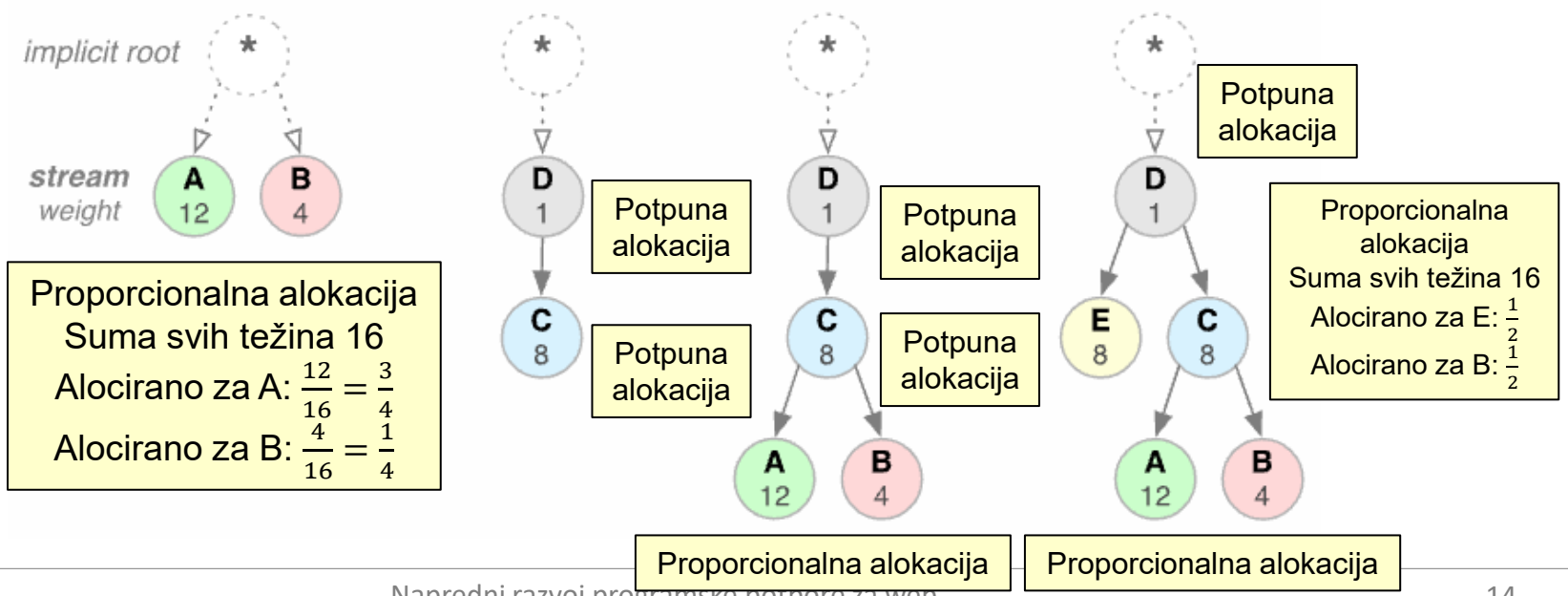
- Važna pozitivna svojstva mehanizma HTTP/2 multipleksiranja:
  1. Paralelno i isprepleteno slanje više zahtjeva bez blokiranja.
  2. Paralelno i isprepleteno slanje više slanje odgovora bez blokiranja.
  3. Uporaba jedne TCP veze za paralelnu isporuku više zahtjeva i odgovora.
  4. Kraće vrijeme učitavanja stranice uklanjanjem nepotrebnih kašnjenja i poboljšanjem korištenja raspoloživog mrežnog kapaciteta.

HTTP/2 multipleksiranje (4)



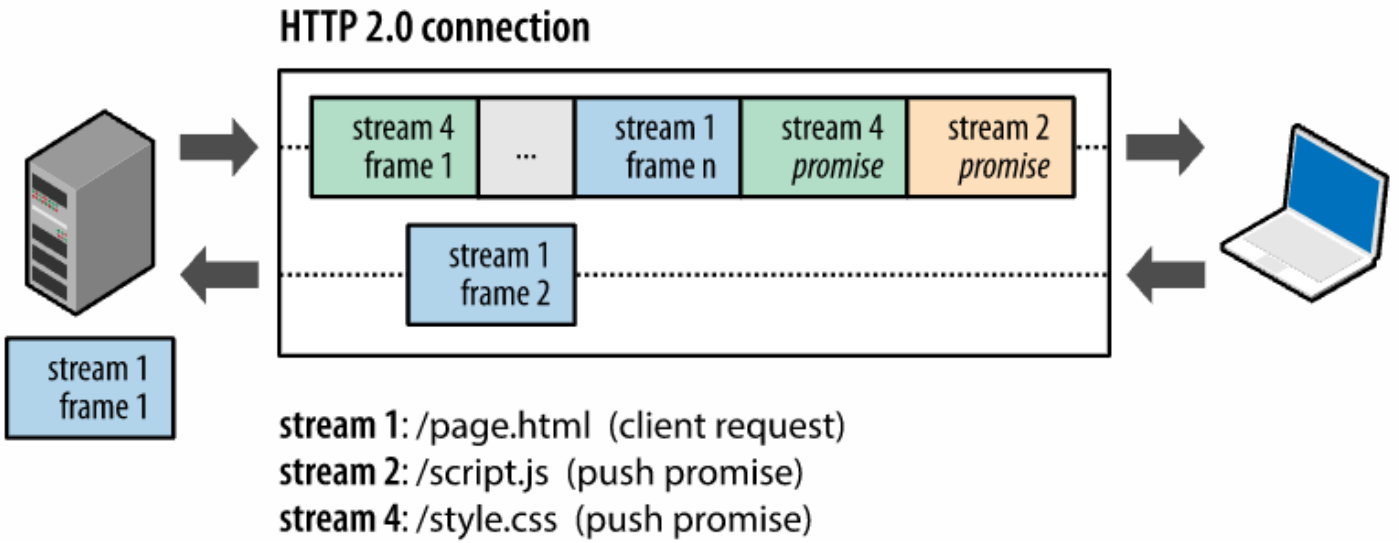
HTTP/2 prioritet toka

- HTTP/2 poruka dijeli se na proizvoljno mnogo pojedinačnih okvira, a okviri iz više tokova se multipleksiraju → **redoslijed okvira postaje važan za performanse**
- Stoga se definira razina prioriteta i **međusobna ovisnost tokova** (***stream prioritization***):
  - Svakom toku može se dodijeliti cjelobrojna težina prioriteta između 0 i 255
  - Svakom toku može se dati izričita ovisnost o drugom toku
- Izgrađuje se **stablo prioriteta** (***prioritization tree***)
  - Proporcionalno udaljenosti od čvora stabla i dodijeljenoj vrijednosti prioriteta poslužitelj alocira računalne resurse, memoriju, procesorsko vrijeme i propusnost mreže. Ovisnosti i težine izražavaju samo transportnu prednost, ne jamče određen redoslijed slanja.



HTTP/2 Server Push

- Poslužitelj može poslati više odgovora na jedan zahtjev klijenta.
  - Osim prvog odgovora, poslužitelj „gura” (*push*) dodatne odgovore bez novih zahtjeva klijenta.
  - Temeljem prvog zahtjeva poslužitelj anticipira sljedeće zahtjeve klijenta. Posljedica je ubrzavanje prijenosa resursa.
  - Poslužitelj prvo šalje **PUSH_PROMISE okvir** da bi „signalizirao” svoju namjeru slanja daljnjih tokova. PUSH_PROMISE okvir sadržava samo zaglavlje HTTP poruke. Podaci (DATA okviri) šalju se kasnije. PUSH_PROMISE mora biti zaprimljen da bi se izbjegli klijentski zahtjevi za istim resursima.
  - Ako je resurs već u privremenoj memoriji klijenta (*cache*) on može odbiti *server push* slanjem RST_STREAM okvira. Ovaj mehanizam nije moguć u HTTP/1.x.

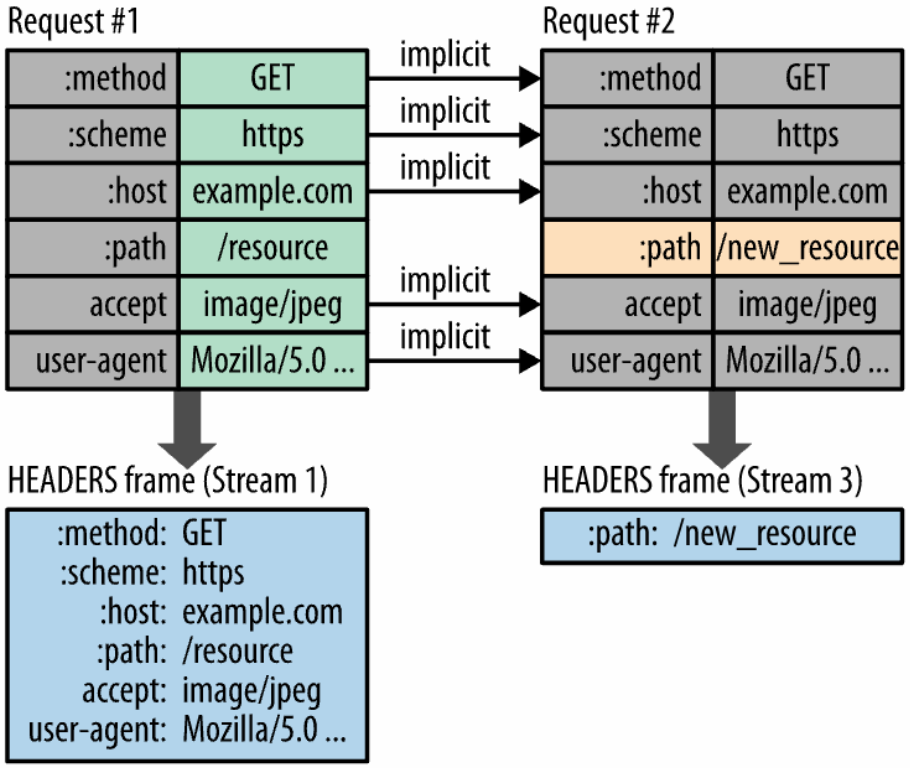


Komprimiranje zaglavlja (1)

- HTTP/1.x prenosi nekodiran *plain text*
  - Svaki HTTP prijenos sadrži skup zaglavlja koja opisuju preneseni resurs i njegova svojstva. U HTTP/1.x, ti se metapodaci uvijek šalju kao običan tekst i dodaju od 500-800 bajtova *overheada* po *requestu*, a ponekad i više ako se koriste HTTP kolačići.
- Kako bi se poboljšale performanse HTTP/2 uvodi **komprimiranje zaglavlja zahtjeva** (*HPACK compression, Header Compression*) korištenjem dvije metode koje zajedno smanjenju potrebu za količinom podataka koja se mora prenijeti:
  1. **Kompresija podataka**
     - Huffmanovo kodiranje zaglavlja.
  2. **Izbjegavanje višestrukog prijenosa podataka**
     - Klijent i poslužitelj uspostavljaju i kontinuirano aktualiziraju indeksiranu tablicu prenesenih polja zaglavlja. Ovime se uspostavlja zajednički kontekst kompresije i onemogućuje se ponovni prijenos podataka koji su već prethodno poslani.

Komprimiranje zaglavlja (2)

- Dodatna optimizacija:
  - **Tablice zaglavlja**:
    1. **Statička tablica** je unaprijed popunjena i sadržava popis uobičajenih polja HTTP/2 zaglavlja koja će vjerojatno koristiti veze i klijenti.
    2. **Dinamička tablica** je u početku prazna i ažurira se na temelju izmjenjenih vrijednosti unutar određene veze.

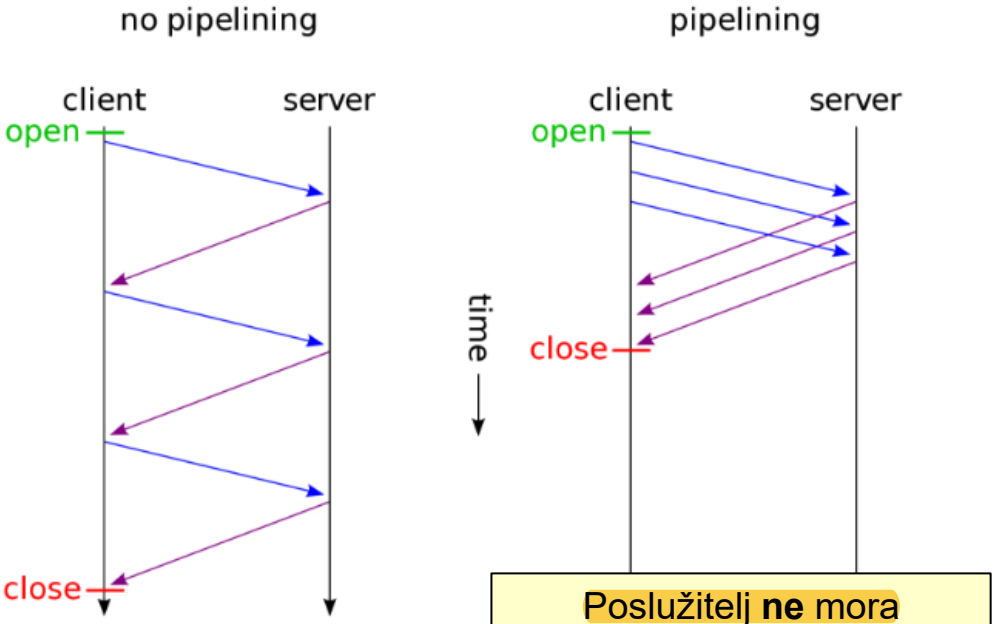


Http2 *pipelining* (1)

- Http pipelining je mehanizam u kojem se više HTTP zahtjeva šalje jednom TCP konekcijom bez čekanja na njihove odgovarajuće odgovore.
  - Kod HTTP/1.x poslužitelj mora održavati ispravan poredak reda odgovora (***response queuing***). Jednak redoslijedu zahtjeva klijenta.
  - To dovodi do ***Head-of-line-blocking*** ograničenja.

Head-of-line-blocking

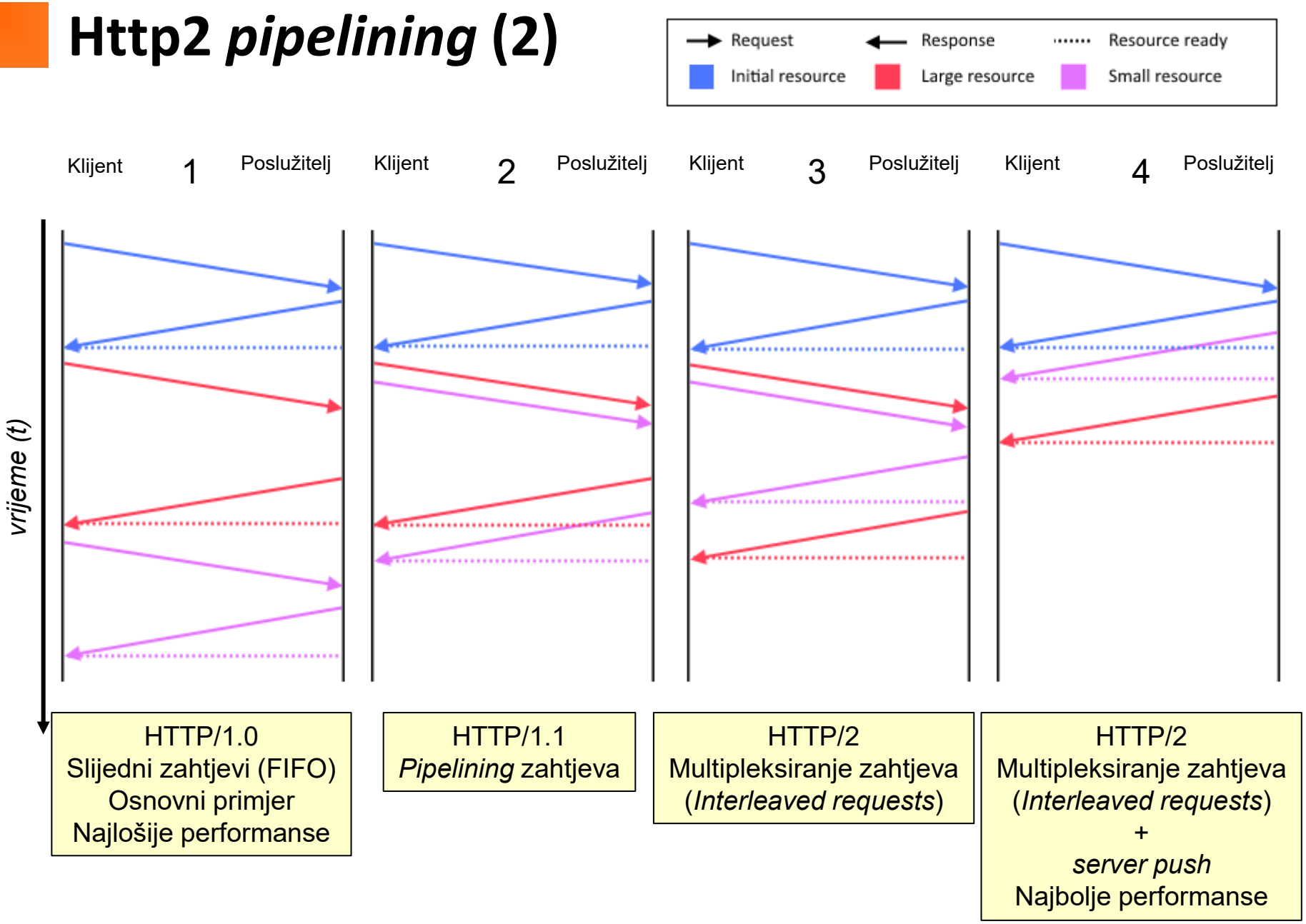
Neka klijent prvo šalje zahtjev **A** koji je složen i zahtijeva znatne resurse poslužitelja, i nakon toga zahtjev **B** koji je procesorski jednostavan i ne zahtijeva znatne računalne resurse. Poslužiteljsko računalo, koji je u stanju riješiti nekoliko zahtjeva odjednom, obraditi će zahtjev **B** vrlo brzo, ali ne može poslati odgovor jer čeka na kraj obrade zahtjeva **A** kako bi poštivao redoslijed zahtjeva. Na taj način spori zahtjevi postaju ograničenje svih sljedećih zahtjeva.



Poslužitelj u odgovoru mora poštivati redoslijed kojim su zaprimljeni zahtjevi

Poslužitelj **ne** mora poštivati redoslijed zahtjeva. Zahtjevi su označeni ID-om toka. *Head-of-line-blocking* nije moguć na OSI razini HTTP.

Http2 *pipelining* (2)



Napad umetanjem zaglavlja

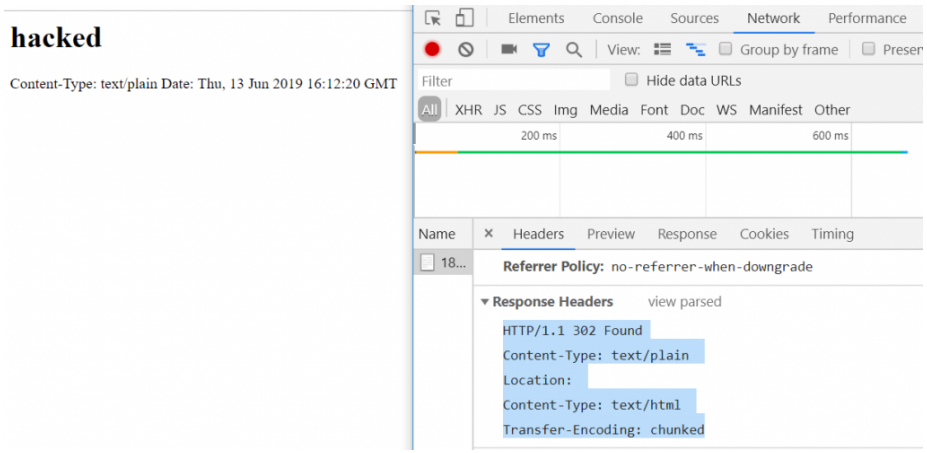
- Napad umetanjem zaglavlja (*header injection attack*)
  - Napadač namjerno manipulira zaglavljima odgovora i ubacuje **(*header injection*)** znakove u zaglavlje HTTP odgovora poslužitelja **(*response splitting attack*)**.
  - Napadač umeće novi odgovor i mijenja vrijednost *Location headera* HTTP poruke

http://example.com/page1%0d%0aLocation:vulnweb.com

```
HTTP/1.1 302 Found
Content-Type: text/plain
Location: \r\n
Content-Type: text/html \r\n\r\n

<html><h1>hacked!</h1></html>
Content-Type: text/plain
Date: Thu, 13 Jun 2019 16:12:20 GMT
```

  - Kao rezultat ubacivanja HTTP zaglavlja CRLF znakovi umetnuti su u odgovor i nakon njih slijedi novo Location: zaglavlje. Web preglednik (ovisno o vrsti i konfiguraciji) preusmjerit će korisnika na napadačevu stranicu (ovdje: vulnweb.com)
  - Slično XSS napadu
  - **HTTP/2 onemogućuje ovakav napad**



Sigurnosni nedostaci HTTP/2 (1)

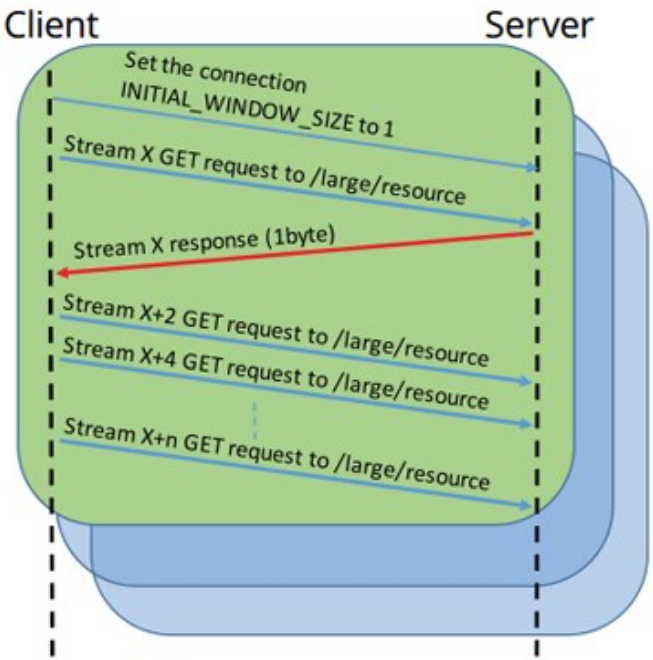
- Nije sve „ružičasto”, i HTTP/2 ima određene sigurnosne nedostatke – **sigurnosne ranjivosti** – koji omogućuju zlonamjerne napade na poslužitelj i neovlašteni pristup podacima.

- Sigurnosni nedostaci HTTP/2 protokola:
  1. **Sporo čitanje** (*Slow Read*)
  2. **HPACK bomba** (*HPACK bomb*, *HPACK attack*)
  3. **Napad ciklusa ovisnosti** (*Dependency Cycle Attack*)
  4. **Zloupotreba multipleksiranja toka** (*Stream Multiplexing Abuse*)

Sigurnosni nedostaci HTTP/2 (2)

1. **Sporo čitanje** (*Slow Read*)
   - Sporo čitanje je napad na poslužitelja sličan Slowloris DDoS napadu, a poziva se na malicioznog klijenta koji čita odgovore pristigle od poslužitelja veoma sporo, čime nepotrebno zauzima resurse poslužitelja i namjerno zadržava otvorenu TCP vezu.

Slowloris DDoS

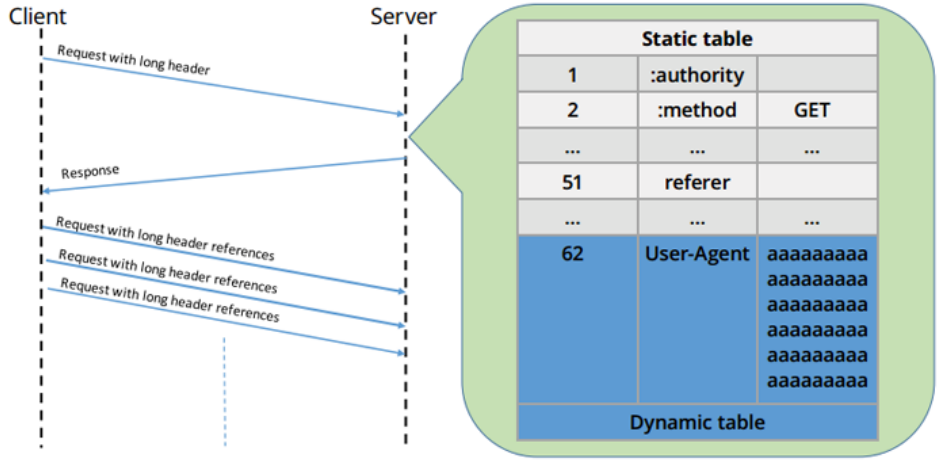


Napad se odvija u 4 koraka:

1. Napadač otvara višestruke veze prema napadnutom poslužitelju slanjem višestrukih djelomičnih zaglavlja HTTP zahtjeva.
2. Napadnuti poslužitelj otvara dretvu za svaki dolazni zahtjev, s namjerom zatvaranja dretve kada se veza završi. Kako bi bio učinkovit, ako veza traje predugo, poslužitelj će prekinuti dretve koje traju vrlo dugo, oslobađajući resurse za sljedeći zahtjev.
3. Napadač želi spriječiti poslužitelja u zatvaranju dretve, stoga povremeno šalje djelomična HTTP zaglavlja zahtjeva kako bi održao zahtjev aktivnim: “Još sam tu! Samo sam spor, molim te pričekaj me.”
4. Poslužitelj nikada ne može otpustiti nijednu otvorenu djelomičnu vezu dok čeka završetak zahtjeva. Nakon što sve dostupne dretvu budu zauzete poslužitelj više neće moći odgovoriti na dodatne zahtjeve od ostalih klijenata, što će rezultirati uskraćivanjem usluge.

Sigurnosni nedostaci HTTP/2 (3)

2. **HPACK bomba** (*HPACK bomb, HPACK attack*)
   - Napad na kompresijski sloj sličan napadu sa tzv. ZIP bombom ("dekompresijska" bomba)
   - Napadač šalje male i naizgled obične poruke, koje se nakon raspakiravanja sadržaja mogu pretvoriti u gigabajte podataka na poslužitelju
   - Na taj način troši se poslužiteljska memorija, usporava rad ili uskraćuju servisi napadnutog poslužitelja



Napadač u dinamičku tablicu zaglavlja (*dynamic table*) umeće polje zaglavlja čija je veličina identična veličini HPACK elementa.

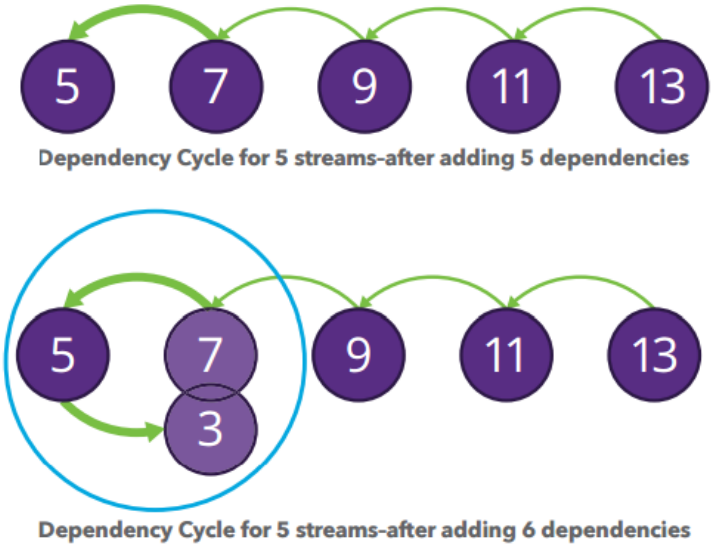
Npr. neka je veličina zaglavlja 4kB i iste veličine kao cijela tablica kompresije. Pretpostavimo da napadač otvara nove HTTP/2 tokove podataka (dozvoljeno više puta, do 16k referenci zaglavlja).

To može dovesti do velikog omjera kompresije od 4096 ili više, što znači da se 16 kB podataka može dekomprimirati u 64 MB podataka na ciljnom računalu.

Nakon slanja 14 takvih tokova, veza je potrošila 896 MB memorije poslužitelja nakon dekompresije.

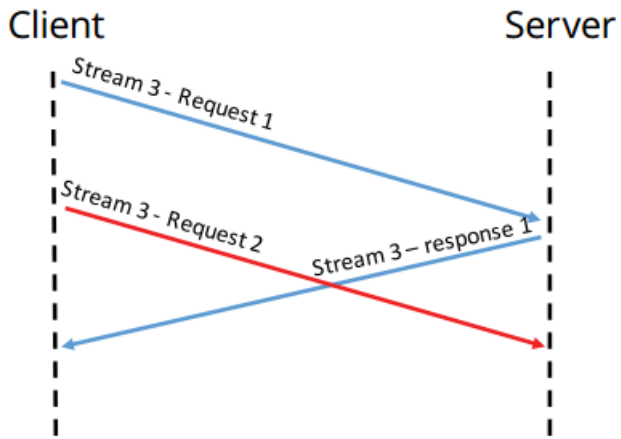
Sigurnosni nedostaci HTTP/2 (4)

3. **Napad ciklusa ovisnosti** (*Dependency Cycle Attack*)
   - Ovaj napad koristi mehanizme kontrole protoka koje u HTTP/2 protokolu služe za optimizaciju mreže.
   - HTTP/2, kao jedno od unaprjeđenja, uveo je i stvaranje prioritizacije i zavisnosti između paketa/podataka koji se šalju preko veze.
   - Međutim, moguće je implementirati krug ovisnosti kao petlju (***dependency cycle***), odnosno prisiljavajući poslužitelj u beskonačnu petlju, te tako izazvati poslužiteljski DoS (*Denial of Service*) napad ili pokrenuti zlonamjeran kôd.



Sigurnosni nedostaci HTTP/2 (5)

4. **Zloupotreba multipleksiranja toka** (*Stream Multiplexing Abuse*)
   - U ovom nedostatku napadač nastoji iskoristiti slabosti HTTP/2 protokola te promijeniti funkcionalnost multipleksiranja preko jedne veze te tako dovesti poslužitelja u stanje nemogućnosti odgovora za prave klijente ili pak potpunog rušenja poslužitelja.



Stream reuse attack diagram

Zaključak

- Zašto koristiti HTTP/2? Ukratko: zbog povećanja performansi i sigurnosti.
- **Kompresija podataka**
  - HTTP/2 nudi ugrađenu kompresiju zaglavlja zahtjeva (HPACK). Suvremene web aplikacije obično prihvaćaju niz različitih zaglavlja, poput autorizacije, podataka o klijentu Iako kompresija ovih podataka možda neće imati velike razlike za jedan zahtjev, postoji mnogo podataka poslanih preko mreže koji se spremaju pri komprimiranju u aplikacijama s velikim prometom. HTTP/1.1 prema zadanim postavkama ne komprimira zaglavlja.
- **Binarni protokol**
  - HTTP/2 je binarni, a HTTP/1.1 tekstni. **Pojednostavljena provedba naredbi, više se ne mogu pogrešno interpretirati zbog korištenja tekstnog formata**. Preglednici koji podržavaju HTTP/2 pretvorit će tekstne naredbe u binarne prije slanja.
- **Sigurnost**
  - **Napadač više ne mogu manipulirati zaglavljima odgovora i ubaciti (*header injection*) znakove u HTTP *response* poslužitelja (*response splitting attack*)**. Tipično, napadač umeće novi odgovor i mijenja vrijednost *Location headera*.